REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 11-05-2015	2. REPORT TYPE Final Report	3. DATES COVERED (From - To) 5-Jun-2011 - 4-Aug-2014

4. TITLE AND SUBTITLE Final Report for the DARPA ADAMS Project	5a. CONTRACT NUMBER
	5b. GRANT NUMBER W911NF-11-C-0215
	5c. PROGRAM ELEMENT NUMBER 1M30BM
6. AUTHORS V.S. Subrahmanian	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES AND ADDRESSES University of Maryland - College Park 3112 Lee Building College Park, MD 20742 -5141	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 60132-NS-DRP.8

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
The principal goal of the Anomaly Detection Engine for Networks (ADEN) was to identify malicious users within a network. We took the word "network" to broadly refer to corporate and government "intranets", as well as networks of users in online communities such as Wikipedia and Slashdot whose goal is to provide correct information to end users. Malicious users within such online communities also constitute a threat inside those networks.
During this project, we worked on 5 different data sets involving insider threat and malicious users. These data sets

15. SUBJECT TERMS
networks, detection, malicious users

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Venkatramanan Subrahmanian
UU	UU	UU	UU		19b. TELEPHONE NUMBER 301-405-6724

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

Report Title

Final Report for the DARPA ADAMS Project

ABSTRACT

The principal goal of the Anomaly Detection Engine for Networks (ADEN) was to identify malicious users within a network. We took the word “network” to broadly refer to corporate and government “intranets”, as well as networks of users in online communities such as Wikipedia and Slashdot whose goal is to provide correct information to end users. Malicious users within such online communities also constitute a threat inside those networks.
During this project, we worked on 5 different data sets involving insider threat and malicious users. These data sets included a CERT data set, a “Vegas” data set, a Wikipedia data set, a Slashdot data set, and the “BAIT” data set that learned behaviors distinguishing real benign users from malicious ones. Because of the varied nature of these data sets, there were different techniques developed.
We worked with open source Wikipedia and Slashdot data sets under the initial impression that finding vandals and trolls in such data would be easy. Though this proved to not be the case, we were eventually able to predict vandals on Wikipedia with over 90% accuracy, using a novel mix of network and language analytics. We were also able to significantly improve both the accuracy and run-time of troll detection within Slashdot.
We also investigated the problem of predicting insiders on data provided by CERT where we achieved high accuracy, finding almost all insider threats injected by CERT. We did this by defining a “codebook” consisting of an extensive set of features used for such predictions.
The most important results were, however, on the BAIT data set that included behaviors of 795 anonymous users, a very small percentage of whom (under 1%) were malicious users with the goal of exfiltrating data. We developed a suite of algorithms to perform classification in highly imbalanced data sets as is likely to be the case with real insider threat detection situations. From the data on the usage of these 795 users, we learned the following important things.
• Malicious users were more active and chose to do ”nothing” significantly less times than benign users.
• Malicious users fetched significantly more ”sensitive information than benign users.• Though malicious users appeared to save more data to removable devices than benign users, these differences were not found to be statistically significant in our study. These findings vary slightly from findings in previous studies which were found to be statistically significant.
• Malicious users edited the data slightly less compared to benign players users. However, these differences also were not found be statistically significant. These findings also vary from findings previously reported in the literature which showed the difference to be statistically significant.
• Malicious users sent significantly more information out of the organization than benign users.
• Malicious users fetched significantly less un-sensitive data in contrast to the benign players.
In addition, we developed novel new notions of collaboration graphs, together with wavelet based time series analysis of these graphs in order to identify anomalous behaviors. We also developed tag-cloud based algorithms and dashboards to explain why particular users are classified as being suspicious.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

<u>Received</u> <u>Paper</u>

TOTAL:

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received	Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received	Paper

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received	Paper

TOTAL:

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received		Paper
05/11/2015	2.00	Amos Azaria, Ariella Richardson, Sarit Kraus, V.S. Subrahmanian. Behavioral Analysis of Insider Threat: A Survey and Bootstrapped Prediction in Imbalanced Data, IEEE Transactions on Computational Social Systems (06 2014)

TOTAL: **1**

Number of Manuscripts:

Books

Received Book

TOTAL:

Received Book Chapter

TOTAL:

Patents Submitted

Patents Awarded

Awards

Graduate Students

NAME	PERCENT_SUPPORTED	Discipline
Noseong Park	0.50	
Anshul Sawant	0.50	
Chanhyun Kang	0.50	
FTE Equivalent:	**1.50**	
Total Number:	**3**	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
Francesca Spezzano	0.75
Edoardo Serra	0.13
Michael Ovelgoenne	0.65
FTE Equivalent:	**1.53**
Total Number:	**3**

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
V.S. Subrahmanian	0.08	
FTE Equivalent:	**0.08**	
Total Number:	**1**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	0.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:......	0.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):......	0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:......	0.00

Names of Personnel receiving masters degrees

NAME	
Total Number:	

Names of personnel receiving PHDs

NAME	
Total Number:	

Names of other research staff

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Sub Contractors (DD882)

Inventions (DD882)

Scientific Progress

Technology Transfer

ADEN: ACTIVITY DETECTION ENGINE FOR NETWORKS

2015

FINAL REPORT FOR THE DARPA ADAMS PROJECT

SUBMITTED TO ARMY RESEARCH OFFICE

CONTRACT NUMBER W911NF-11-C-0215

Prepared by

Dept. of Computer Science & UMIACS
University of Maryland

College Park, MD 20742.

PI: V.S. Subrahmanian

vs@cs.umd.edu

301-405-6724

ABSTRACT

The principal goal of the Anomaly Detection Engine for Networks (ADEN) was to identify malicious users within a network. We took the word “network” to broadly refer to corporate and government “intranets”, as well as networks of users in online communities such as Wikipedia and Slashdot whose goal is to provide correct information to end users. Malicious users within such online communities also constitute a threat inside those networks.

During this project, we worked on 5 different data sets involving insider threat and malicious users. These data sets included a CERT data set, a “Vegas” data set, a Wikipedia data set, a Slashdot data set, and the “BAIT” data set that learned behaviors distinguishing real benign users from malicious ones. Because of the varied nature of these data sets, there were different techniques developed.

We worked with open source Wikipedia and Slashdot data sets under the initial impression that finding vandals and trolls in such data would be easy. Though this proved to not be the case, we were eventually able to predict vandals on Wikipedia with over 90% accuracy, using a novel mix of network and language analytics. We were also able to significantly improve both the accuracy and run-time of troll detection within Slashdot.

We also investigated the problem of predicting insiders on data provided by CERT where we achieved high accuracy, finding almost all insider threats injected by CERT. We did this by defining a “codebook” consisting of an extensive set of features used for such predictions.

The most important results were, however, on the BAIT data set that included behaviors of 795 anonymous users, a very small percentage of whom (under 1%) were malicious users with the goal of exfiltrating data. We developed a suite of algorithms to perform classification in highly imbalanced data sets as is likely to be the case with real insider threat detection situations. From the data on the usage of these 795 users, we learned the following important things.

- Malicious users were more active and chose to do ”nothing” significantly less times than benign users.
- Malicious users fetched significantly more ”sensitive information than benign users.

- Though malicious users appeared to save more data to removable devices than benign users, these differences were not found to be statistically significant in our study. These findings vary slightly from findings in previous studies which were found to be statistically significant.
- Malicious users edited the data slightly less compared to benign players users. However, these differences also were not found be statistically significant. These findings also vary from findings previously reported in the literature which showed the difference to be statistically significant.
- Malicious users sent significantly more information out of the organization than benign users.
- Malicious users fetched significantly less un-sensitive data in contrast to the benign players.

In addition, we developed novel new notions of collaboration graphs, together with wavelet based time series analysis of these graphs in order to identify anomalous behaviors. We also developed tag-cloud based algorithms and dashboards to explain why particular users are classified as being suspicious.

I. INTRODUCTION & MAIN CONTRIBUTIONS

The overall goal of the ADEN project was to detect malicious users within a network. Malicious users within networks can take several different forms. A few examples that we investigated are given below.

- Individuals within an organization can be malicious in the sense that they steal data (such as Edward Snowden) and leak it to external parties such as Wikileaks or a foreign nation state.
- Individuals within a collaborative network such as Wikipedia may make attempts to derail the overall goal of the network. For instance, "vandals" on Wikipedia may make attempts to compromise the integrity of the data within Wikipedia, rendering it less trustworthy for all.
- Individuals within a social network such as Slashdot rely on comments by others in order to gauge who to trust and who not to trust. Users called "trolls" try to derail such a worthy agenda by inserting misinformation, but concurrently trying to boost their status within the organization via a set of dubious methods.

These are just three examples of the types of problems that necessitate the need to identify malicious actors within networks. All three problems are closely related to one another because it is behavior that separates malicious users from benign ones.

In order to address the overall goal of the ADAMS project, we worked with 5 data sets in all.

- Wikipedia Data: We worked with two tranches of Wikipedia data with the goal of predicting who in the Wikipedia author network is a vandal. Initially, we started with a natural language based approach with a collection of about 895 Wikipedia documents that were annotated. Later, we developed a combined NLP and network based approach to prediction of vandals on Wikipedia – which yields over 90% predictive accuracy, beating past work by almost 20%.
- Slashdot Data: We worked with one tranch of Slashdot data. In Slashdot, users can leave comments about other users, marking them with positive and negative links. We developed algorithms to use such explicit network information in order to identify troll like behavior. Our Troll Identification Algorithm includes a novel step to "declutter" a network, getting rid of irrelevant factors, and focusing the analysis on a portion of the

network likely to contain bad actors. The result is a huge increase in accuracy of prediction of such bad actors by almost a 3-fold factor.

- Behavioral Analysis of Insider Threat (BAIT) Data: One major flaw with the 2 data sets generated by the ADAMS project is that the data did not involve real malicious users. For instance, CERT generated synthetic attacks crafted by CERT personnel. A similar situation was true with the SureView data. In BAIT, we set up a controlled experiment on Amazon Mechanical Turk with 795 real humans playing a synthetic insider threat game. Users would target the exfiltration of data. Some users were malicious, most were benign. Using sophisticated methods of enhancing training data sets, we were able to learn strategies that real insiders might use to exfiltrate information. Some of these findings confirmed prior hypothesis – but there were a few findings that differed from past work.
- CERT Data: The Computer Emergency Response Team (CERT) generated synthetic data about insiders within a hypothetical organization. Our goal was to identify malicious insiders within that organization. We were able to succeed in detecting 32 out of 37 malicious incidents, leading to a recall of over 86%.
- SureView Data: The SureView data was provided by a major defense contractor. One challenge we faced was that of working at their site with US personnel only. As a consequence, our ability to do well on the SureView data was limited.

Overall, we made the following significant contributions to our understanding of malicious activities in networks, both within a single organization, as well as within virtual organizations (such as the Wikipedia network and Slashdot networks). The specific contributions are as follows:

- Wikipedia Vandal Detection. We study the problem of detecting vandals on Wikipedia before any human reports flagging a potential vandal so that such users can be presented early to Wikipedia administrators. We leverage multiple classical ML approaches, but develop 3 novel sets of features. Our Wikipedia Vandal Behavior (WVB) approach uses a novel set of user editing patterns as features to classify some users as vandals. Our Wikipedia Transition Probability Matrix (WTPM) approach uses a set of features derived from a transitional probability matrix and then reduces it via a neural net auto-encoder to classify some users as vandals. The VEWS approach merges the previous two approaches. Without using any information (e.g. reverts) provided by other users, these algorithms each have over 85% classification accuracy. Moreover, when temporal recency is considered, accuracy goes to almost 90%. We carry out detailed experiments on a data set consisting of about 33K Wikipedia users (including both a black list and a white list of authors) and containing 770K edits. We describe specific behaviors that distinguish between vandals and non-vandals. We show that

VEWS beats ClueBot NG and STiki, the best known algorithms today for vandalism detection. Moreover, VEWS detects far more bots than ClueBot and on average, detects them 2.39 edits before ClueBot. However, we show that the combination of VEWS and ClueBot NG can have a higher accuracy than both.

- Slashdot Troll Detection. We proposed the TIA algorithm that takes any centrality measure and any set of decluttering operations as input parameters, and uses them to iteratively identify the trolls in the Slashdot Network. Using the standard Average Precision measure to capture accuracy of our TIA algorithm, we show that TIA using Signed Eigenvector Centrality and two specific decluttering operations gives us the best result of 51.04%, significantly exceeding the 15.07% when no decluttering operations are performed. Compared to the best existing results on troll detection in Slashdot [3], our algorithm runs 27 times faster on the original network and 35-50 times faster on various sub-networks. Moreover it is able to retrieve about twice as many trolls with a Mean Average Precision which is more than three times as good as past work.
- Behavioral Analysis of Insider Threat. We developed the BAIT (Behavioral Analysis of Insider Threat) framework, in which we conduct a detailed experiment involving 795 subjects on Amazon Mechanical Turk in order to gauge the behaviors that real human subjects follow when attempting to exfiltrate data from within an organization. In the real world, the number of actual insiders found is very small, so supervised machine learning methods encounter a challenge. Unlike past works, we develop bootstrapping algorithms that learn from highly imbalanced data, mostly unlabeled, and almost no history of user behavior from an insider threat perspective. We develop and evaluate 7 algorithms using BAIT and show that they can produce a realistic (and acceptable) balance of precision and recall. Moreover, we learned differences between behavior of real malicious users and that of benign users. Specifically we learned that:
    - The malicious players were more active and chose to do ”nothing” significantly (p <0:01) less times (3:77) than the benign players (6:52).
    - The malicious players fetched significantly (p < 0:05) more ”sensitive information (9:8) than the benign players (8:26).
    - The malicious players appeared to save more data to CD/DVD (1:2) and USB (2:225) compared to the benign players CD/DVD (1:07) and USB(1:82). However, these differences were not found to be statistically significant in our study. These findings vary slightly from findings in previous studies which were found to be statistically significant.
    - The malicious players edited the data slightly less (3:77) compared to the benign players (3:88). However, these differences also were not found be statistically significant. These findings also vary from findings previously reported in the literature which showed the difference to be statistically significant.

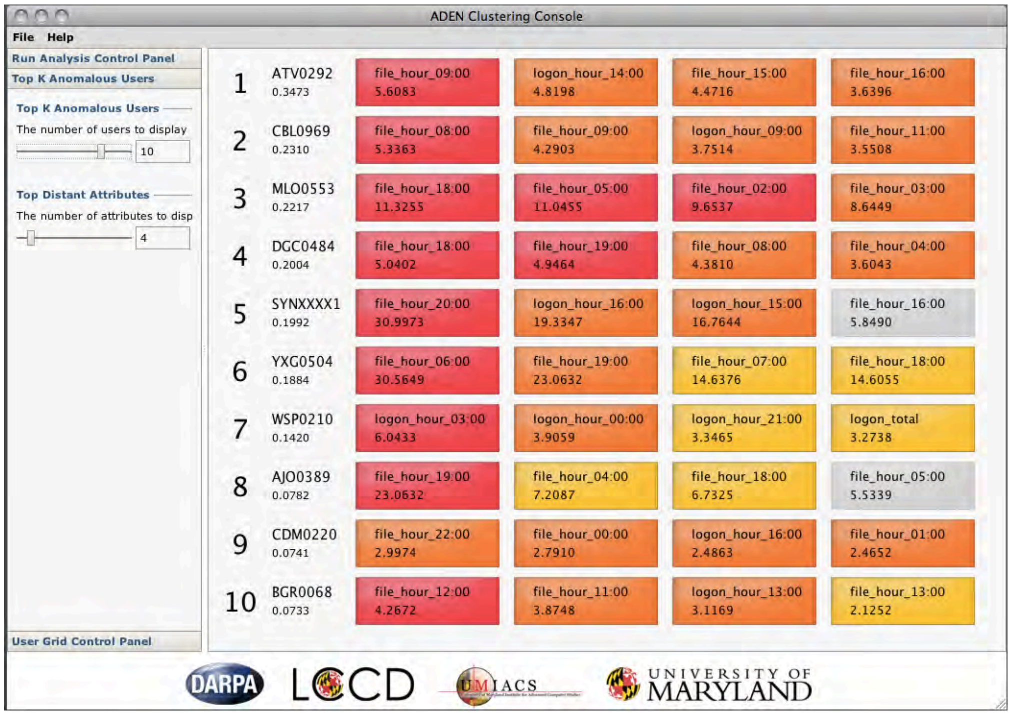
- In addition, we found the following statistically significant differences between the malicious and the benign players. No similar findings have been reported in prior work.
  - The malicious players sent significantly ($p < 0.001$) more information out of the organization with an average of (2:92) times, in contrast to the benign players (0:68).
  - The malicious players fetched significantly ($p = 0:05$) less unclassified data (2:87) in contrast to the benign players (3:40).

Simply put, some of these findings are extremely novel and cast important light on the behavior of real world insiders as compared to synthetic insiders injected somehow by an artificial process.

We now discuss these contributions in the rest of this report.

II. FINDING MALICIOUS USERS IN THE CERT DATA

The initial synthetic datasets provided by CERT are log files of user activity. We developed a codebook of variables that describe the log file data. The codebook contains for example variables for the average number of log on events by hour, the average number of files copied to thumb-drives by hour. In our initial approach (see Figure 1) we generated for every variable averages over the complete time horizon of the dataset and experimented with clustering approaches to identify outliers. Each user was represented by a vector of codebook values. Users that are not in dense regions of the multi-dimensional space spanned by the user vectors have been regarded as outliers. This approach did not satisfactory precision. Despite efforts to handle the dimensionality problem (we experimented with dimensionality reduction) we did not get satisfactory results. The low noise/information ratio showed to be a major problem. Furthermore, users showed to have inhomogeneous behaviors. Non-standard behavior with respect to the total user base seems to be no good indicator for malicious behavior.



Top K Anomalous Users
The number of users to display: 10

Top Distant Attributes
The number of attributes to disp: 4

1	ATV0292 0.3473	file_hour_09:00 5.6083	logon_hour_14:00 4.8198	file_hour_15:00 4.4716	file_hour_16:00 3.6396
2	CBL0969 0.2310	file_hour_08:00 5.3363	file_hour_09:00 4.2903	logon_hour_09:00 3.7514	file_hour_11:00 3.5508
3	MLO0553 0.2217	file_hour_18:00 11.3255	file_hour_05:00 11.0455	file_hour_02:00 9.6537	file_hour_03:00 8.6449
4	DGC0484 0.2004	file_hour_18:00 5.0402	file_hour_19:00 4.9464	file_hour_08:00 4.3810	file_hour_04:00 3.6043
5	SYNXXXX1 0.1992	file_hour_20:00 30.9973	logon_hour_16:00 19.3347	logon_hour_15:00 16.7644	file_hour_16:00 5.8490
6	YXG0504 0.1884	file_hour_06:00 30.5649	file_hour_19:00 23.0632	file_hour_07:00 14.6376	file_hour_18:00 14.6055
7	WSP0210 0.1420	logon_hour_03:00 6.0433	logon_hour_00:00 3.9059	logon_hour_21:00 3.3465	logon_total 3.2738
8	AJO0389 0.0782	file_hour_19:00 23.0632	file_hour_04:00 7.2087	file_hour_18:00 6.7325	file_hour_05:00 5.5339
9	CDM0220 0.0741	file_hour_22:00 2.9974	file_hour_00:00 2.7910	logon_hour_16:00 2.4863	file_hour_01:00 2.4652
10	BGR0068 0.0733	file_hour_12:00 4.2672	file_hour_11:00 3.8748	logon_hour_13:00 3.1169	file_hour_13:00 2.1252

Figure 1: ADEN Clustering Console

From what we learned from the global outlier analysis, we regard completely unsupervised approaches to identifying attackers and attacks to be less promising. For example, learning the predictive power of different types of user behavior is hardly possible. Therefore, we investigated other approaches that require some user input. User inputs are a way to integrate external information.

Based on the experience of the first prototype, we designed a new detector engine. Our key design decisions were:

- Measure anomaly by temporal changes in the behavior of a user
- Address dimensionality problem by directly selecting promising dimensions
- Put more emphasis on user-system interaction: because of the expected high false positive rate of the detection system, the user needs to get actionable information why a user raised suspicion

For our new approach we continue to extract the codebook variables (1st degree variables) from the raw log data (CERT or SureView). Then we derive from these variables so-called 2nd degree variables. These 2nd degree variables encode indicators of insider attacks as well as indications that a user has an increased proneness to conduct attacks.

We reviewed literature on insider attacks to identify what makes a user become an attacker. From the literature review we created a list of symptoms (e.g. professional or private stress) and several indicators for each symptom. In this way we will complement the detection of abnormal behavior with information on not uncommon but worrisome behavior. We think we will make best use of the sparse information in the large pool of data by this combined analysis of symptoms and abnormal behavior.

From the literature review we derived four major indicators that are correlated with the probability that a user starts an insider attack:

- Start of Employment
- Termination of Employment
- Personal Problems (illness/death in family, marital/relationship problems)
- Professional Problems (negative changes at workplace, interpersonal conflicts)

Furthermore, we encode in 2nd degree variables indicators that are directly related to attacks, for example:

- Increased File Access
- Copy of Executable Files to a Computer
- Unusual working hours

Our system raises alerts, when more than a user-defined number of 2nd degree variables indicate an attack. This way, users can configure the sensitivity of the system.

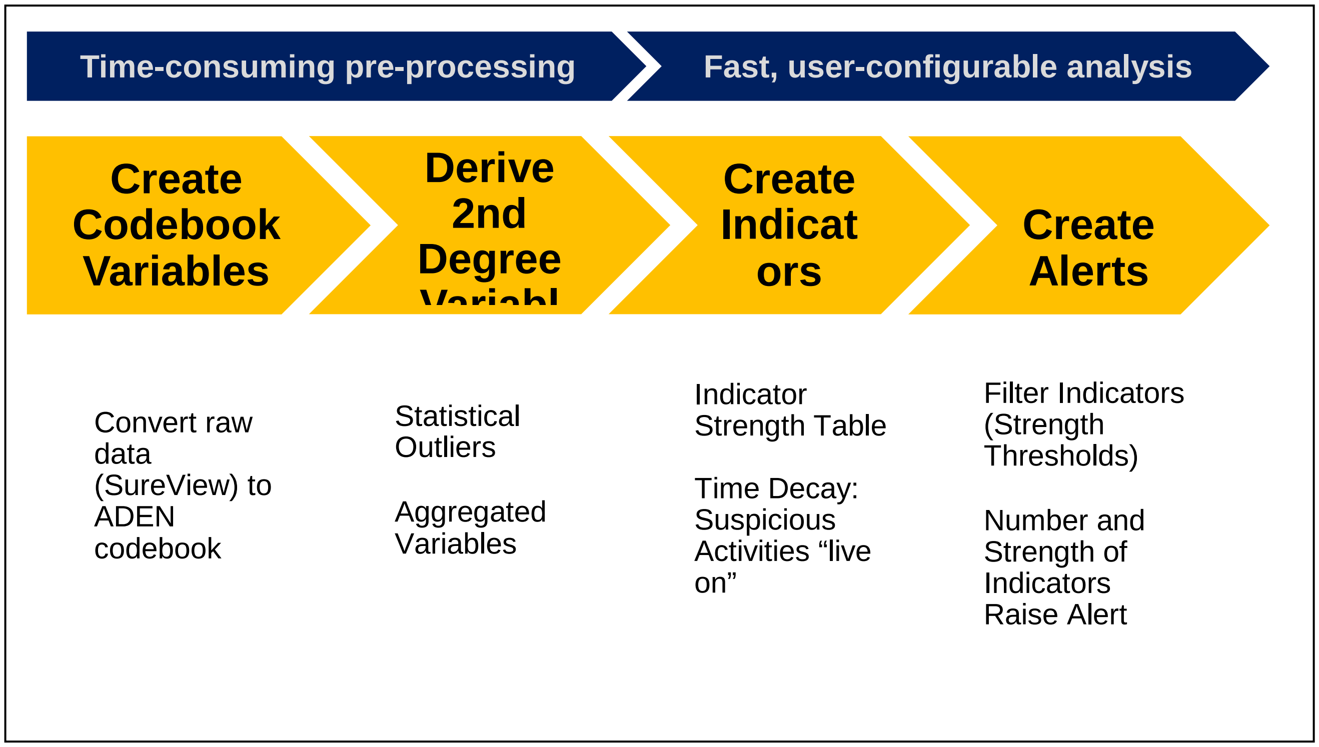

Time-consuming pre-processing
Fast, user-configurable analysis
Create Codebook Variables
Derive 2nd Degree Variabl
Create Indicat ors
Create Alerts
Convert raw data (SureView) to ADEN codebook
Statistical Outliers
Aggregated Variables
Indicator Strength Table
Time Decay: Suspicious Activities "live on"
Filter Indicators (Strength Thresholds)
Number and Strength of Indicators Raise Alert

Figure 2: Process Flow ADEN Detector Engine

Dataset	Alerts	Suspicious Users	Detected Incidents
Cert3v1	44	29	2 / 2
Cert3v2	46	33	2 / 2
Cert4v1	46	25	2 / 3
Cert4v2	139	75	26 / 30

Table 1: Evaluation Results ADEN Detector Engine

Figure 2 shows the analysis process of the ADEN Detector Engine. In the two first steps codebook variables are generated and used to create 2nd degree variable. The next two steps are fast and user-configurable to give the user interactive feedback. This process flow bundles the computational expensive tasks in pre-processing steps, so that the system can provide quick responses to user inputs.

The results of this approach are very promising (see Table 1). We analyzed 4 datasets and identified with standard parameter settings most incidents with a good precision (for this kind of problem).

III. FINDING MALICIOUS USERS IN THE WIKIPEDIA DATA

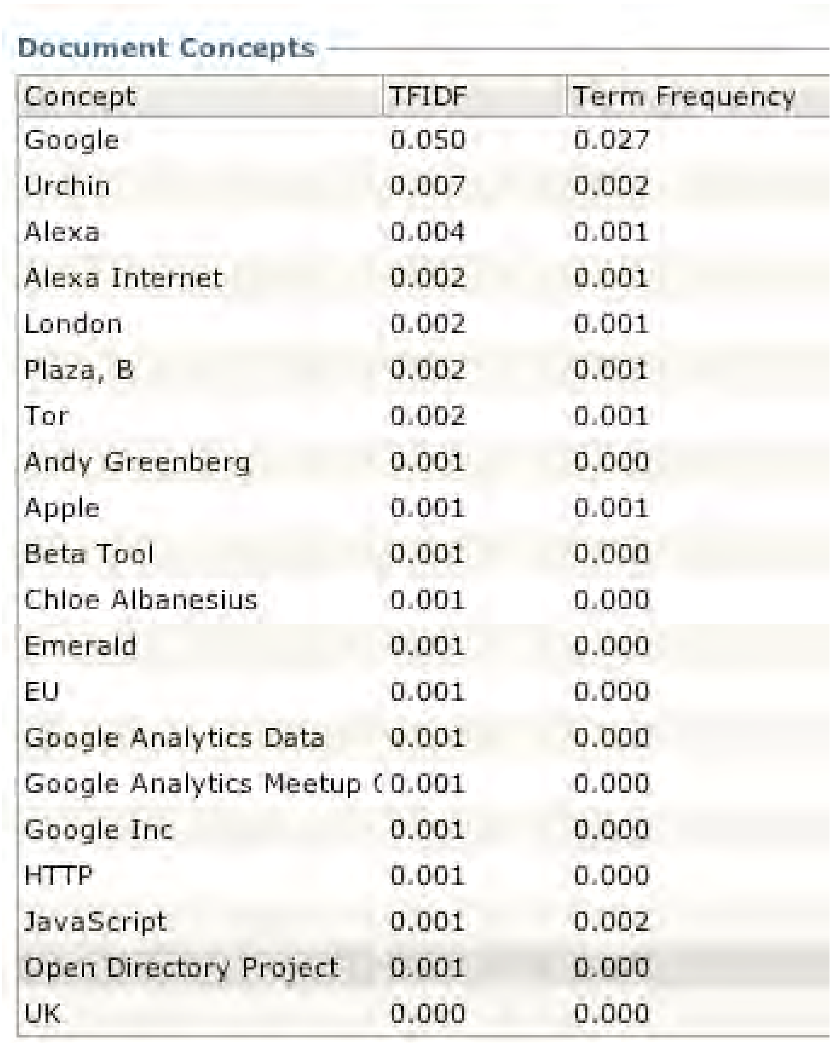
III.A EARLY ATTEMPTS

Our early attempts to find vandals on Wikipedia worked as follows. We identified a small set of Wikipedia documents on 3 topics

- Computers
- Aviation
- Genetics

Each set had a few known malicious users (vandals). We looked at all of the articles and used natural language processing in order to extract a concept histogram, describing the occurrences of different types of concepts in these data sets. The goal was to see if a histogram associated with these concepts would somehow allow us to identify vandals.

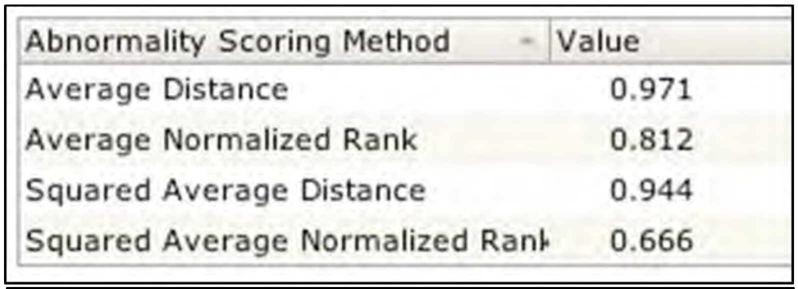
For instance, in our “Computers” data set, we identified 1795 concepts such as those shown in the figure below.



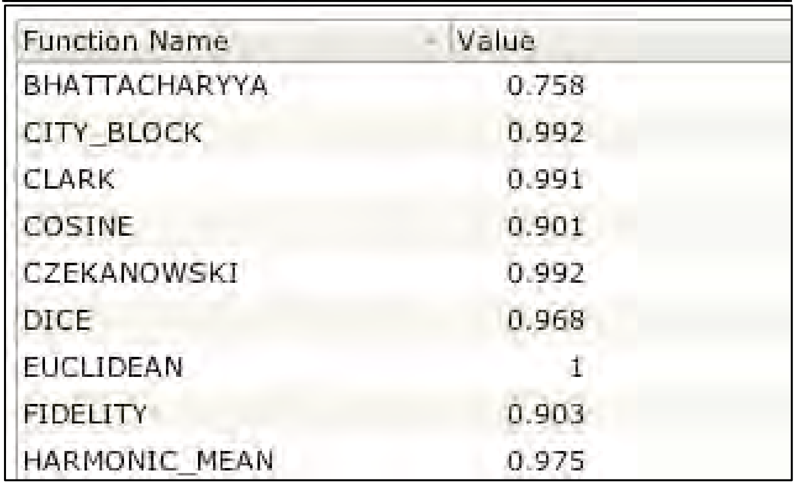
Document Concepts

Concept	TFIDF	Term Frequency
Google	0.050	0.027
Urchin	0.007	0.002
Alexa	0.004	0.001
Alexa Internet	0.002	0.001
London	0.002	0.001
Plaza, B	0.002	0.001
Tor	0.002	0.001
Andy Greenberg	0.001	0.000
Apple	0.001	0.001
Beta Tool	0.001	0.000
Chloe Albanesius	0.001	0.000
Emerald	0.001	0.000
EU	0.001	0.000
Google Analytics Data	0.001	0.000
Google Analytics Meetup (	0.001	0.000
Google Inc	0.001	0.000
HTTP	0.001	0.000
JavaScript	0.001	0.002
Open Directory Project	0.001	0.000
UK	0.000	0.000

We then looked at the concept histogram associated with the posts (on Wikipedia) of each and every user. Thus, each user *u* had an associated histogram *h(u)* and this could be compared with the histogram *H(ALL)* of the entire suite of users. Because we expected individual users who were malicious to behave differently in terms of the content of their posts, we expected significant differences between *h(u)* and *h(ALL)* for malicious users *u.* We used over 15 measures to compare distances between two histograms such as those shown in the figure below.

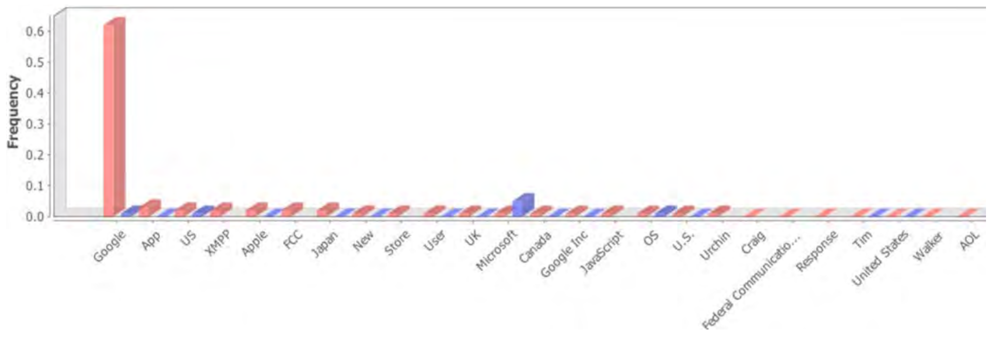


Abnormality Scoring Method	Value
Average Distance	0.971
Average Normalized Rank	0.812
Squared Average Distance	0.944
Squared Average Normalized Ranl	0.666



Function Name	Value
BHATTACHARYYA	0.758
CITY_BLOCK	0.992
CLARK	0.991
COSINE	0.901
CZEKANOWSKI	0.992
DICE	0.968
EUCLIDEAN	1
FIDELITY	0.903
HARMONIC_MEAN	0.975

Initially, this approach paid off well and we were able to find some vandals quickly – an example is shown in the figure below.



However, we quickly determined that the false positive rate was too high and that this approach was too simplistic to work.

III.B VANDAL EARLY WARNING SYSTEM

Undeterred, we went on to develop a much more sophisticated system to identify malicious users in the Wikipedia network. In order to achieve this, we developed three approaches.

Wikipedia Vandal Behavior (WVB) Algorithm. In this approach, we defined a large set of features as follows.

1. Two consecutive edits, slowly (cs): whether or not the user edited the same page consecutively with a gap exceeding 15 mins.

2. Two consecutive edits, very fast (cv): whether or not the user edited the same page consecutively and less than 3 mins passed between the two edits.

3. Consecutive re-edit of a meta-page (crm): num- ber of times that the user re-edited the same meta-page, consecutively.

4. Consecutive re-edit of a non-meta-page (crn): whether or not the user re-edited the same non-meta-page, consecutively.

5. Consecutive re-edit of a meta-page, very fast (crmv): whether or not the user re-edited the same meta-page, consecutively, and less than 3 mins passed between the two edits.

6. Consecutive re-edit of a meta-page, fast (crmf): whether or not the user re-edited the same meta-page, conecutively, and 3 to 15 mins passed between the two edits.

7. Consecutive re-edit of a meta-page, slowly (crms):whether or not the user re-edited the same meta-page, consecutively, and more than 15 mins passed between the two edits.

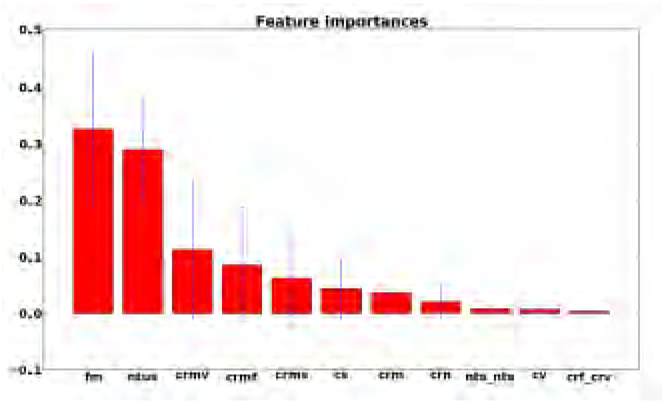
8. Consecutively re-edit fast and consecutively reedit very fast (crf crv): whether or not the following pattern is observed in the user log. The user re-edited the same article within 15 mins, and later re-edited a (possibly different) article and less than 3 mins passed between the second pair of edits.

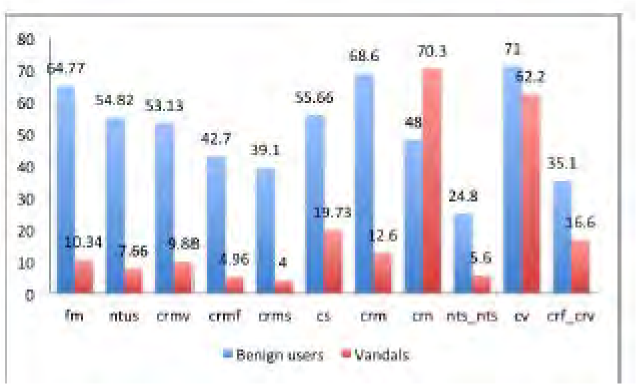
9. First edit meta-page (fm): whether or not the first edit of the user was in a meta-page. This in itself is quite a distinguishing feature, because vandals first edit a non-meta page and benign users first edit a meta-page. Therefore, this becomes quite an important feature for distinguishing the two.

10. Edit of a new page at distance at most 3 hops, slowly (ntus): whether or not the user edited a new page (never edit by him before) p2 which is within 3 hops or less of the previous page p1 that he edited and either p1 or p2's category is unknown9 and the time gap between the two edits exceeds 15 minutes.

11. Edit of a new page at distance at most 3 hops slowly and twice (nts nts): whether or not there are two occurrences in the user log of the following feature edit of a new page at distance at most 3 hops, slowly (nts), i.e. in a pair (p1; p2) of consecutive edits, whether or not the user edited a new page p2 (i.e. never edited before) s.t. p2 can be reached from p1 link-wise with at most 3 hops, and more than 15 mins passed between the edit of p1 and p2.

Using these features, we were able to learn a classifier that separated out vandals from non-vandals on a separate Wikipedia data set consisting of about 30K users in all, split roughly evenly between benign and malicious users. The two figures below show the relative importance of different features in our data set.





Based on these parameters, we see immediately that the features *fm, ntus, crmv, crmf, crms* distinguish very well between malicious users and benign ones.

Wikipedia Transition Probability Matrix Algorithm. We then defined the novel notion of a Wikipedia Probabilistic Transition Matrix.

The Wikipedia Transition Probability Matrix (WTPM) captures the edit summary of the users. The states in WTPM correspond to the space of possible vectors of features associated with any edit pair (p1; p2) carried out by a user u. The intuition behind using WTPM as features for classification is that the transition probability from one state to the other for a vandal may differ from that of a benign user. Moreover, the states visited by vandals may be different from states visited by benign users (for example, it turns out that benign users are more likely to visit a state corresponding to first edit on meta page", as compared to vandals).

We create a compact version of the transition matrix using an auto-encoder. When doing cross-validation, we train the auto-encoder using the training set with input from both benign users and vandals. We then take the value given by the hidden layer for each input as the feature for training a classifier. For predicting output for the test set, we give each test set as input to the auto-encoder and feed its representation from the hidden layer into the classifier. Note that the auto-encoder was trained only on the training set, and the representation for the test set was only derived from this learned model.

The accuracy of SVM for prediction using all the entries in the auto-encoder representation of the WTPM was over 87%.

VEWS Algorithm. The "initial" VEWS algorithm simply combined all the features in both WVB and WTPM and applied SVM. This algorithm achieved an accuracy of 87.82%, slightly higher than that of WTPM.

We then extended VEWS with natural language features captured by existing systems such as ClueBot and STiki. The resulting classifier achieved classification accuracy of over 92%, significantly higher than those achieved by prior work (ClueBot and STiki achieved classification accuracy in the low 70% range).

IV. FINDING MALICIOUS USERS ON SLASHDOT NETWORKS

Unlike many networks, Slashdot is an example of a *signed* social network in which edges are labeled with positive or negative signs. For instance, if an edge from Mary to John is positive, this means that Mary trusts John, while if it is negative, it means that Mary distrusts John. We showed that it is possible to analyze such trust-distrust relationships within signed social networks in order to find malicious users. It is important to note that signed social networks can also be inferred from email traffic. For instance, if person *a* says positive things about a person *b* in his emails, then we can place a positive link from *a* to *b*, but if *a* says negative things about *b*, then we can place a negative link from *a* to *b*. Thus, the methods used in this work apply also to insider threat within organizations where such links can be inferred through appropriate NLP methods.

Formally, a signed social network is a *weighted graph G=(V,E,w)* where *V* is a set of vertices (users), *E* is a set of edges, and *w* is a mapping that assigns either "+" or "-" to each edge. In order to predict malicious users (or trolls) on Slashdot, we built on work by Kunegis et al and considered several independent variables associated with each user *v*.

- Freak(v). This is merely the number of users who are negative about *v*, i.e. $Freak(v) = |\{u \mid (u,v) \in E \ \wedge w(u,v) = '-'\}|$.

- FMF(v) – Fans Minus Freaks. This is merely the difference between the number of people who like *v* and the number who don't, i.e. $FMF(v) = |\{u\,|(u,v) \in E \ \wedge \ w(u,v) =' +'\}| - |\{u\,|(u,v) \in E \wedge w(u,v) =' -'\}|$.
- MPR(v) – Modified Page Rank. The Modified PageRank of *v* is computed as follows
  - Find the PageRank of *v* w.r.t. the subgraph of *G* that only contains positive edges
  - Find the PageRank of *v* w.r.t. the subgraph of *G* that only contains negative edges
  - Subtract the second quantity from the first
- SSR(v) – Signed Spectral Rank. Signed Spectral Ranking (SSR) improves upon PageRank by taking edge signs into account. It is computed by taking the dominant left eigenvector of the signed adjacency matrix associated with the graph *G*.
- NR(v) – Negative Rank. The Negative Rank of *v* is obtained by:
  - Computing SSR(v)
  - Computing PR(v)
  - Subtracting β times PR(v) from SSR(v) where β is a variable parameter.
- SEC(v) – Signed Eigenvector Centrality. The Signed Eigenvector Centrality of *v* is the value assigned to *v* by the dominant eigenvector of the signed adjacency matrix of *G*.

In addition to these features, we considered a set of other features as well. The general idea behind our work here was to identify those users with the lowest measures of centrality and consider them to be trolls. For instance, users with the lowest FMF(v) values have the smallest numbers of fans and the largest numbers of freaks, suggesting that most people dislike them, which in turns suggests that they are not benign users (as people may not be opposed to benign users).

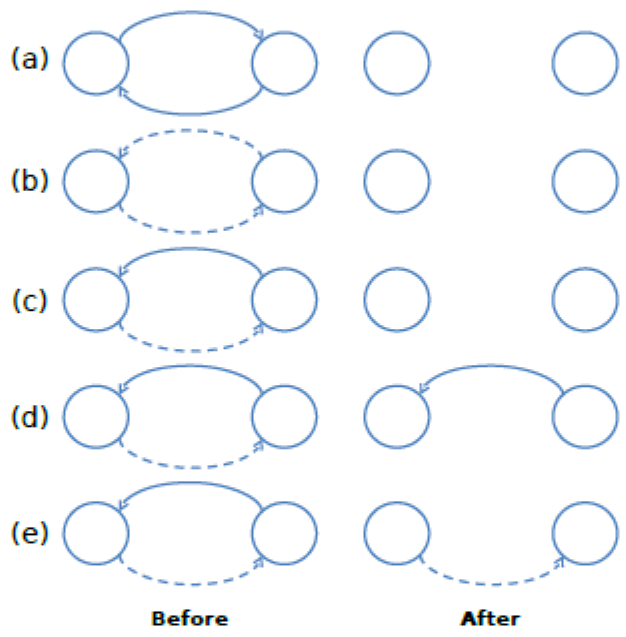
However, this idea did not work. The main reason for this is that in large data sets, the number of edges is very large. Moreover, malicious users operate in "botnet" style environments. They do a number of things such as:

- *Individual malicious peers*. Malicious users always present bad behavior, and hence receive negative links from good users. These are relatively stupid malicious users who should not be difficult to detect, say by using Freaks centrality.
- *Malicious collectives*. Malicious users endorse other malicious users. In this case, a malicious user's score may increase due to the presence of a bunch of positive incoming links.
- Camouflage behind good transactions. Malicious users can cheat some benign users to vote positively for them. This happens, for instance, when malicious users endorse a benign user who, out of courtesy, endorses them back.

- Malicious spies. There are two kinds of malicious users: some of them act as in threat models B and C, while the others (called spies) make benign users to vote positively for them, and assign positive value only to bad nodes.
- Camouflage behind judgements. The strategy in this case is to assign negative value to good users. Then, this can cause the decrease of rank for good peers, and, consequently, the increase of malicious user's rank.

By "gaming" the system in these ways, malicious users are able to significantly "clutter" the network with spurious users and links in an effort to amplify their own centrality scores which measure trustworthiness.

In order to address this, we developed a space of decluttering operations whose goal is to remove a bunch of irrelevant edges. We defined 5 possible decluttering operations shown in the figure below.



Formally, the 5 operations are as follows, assuming some initial labeling of nodes as benign or malicious. One way to achieve this (initially) using a signed centrality measure *C* (any of those listed earlier) is to apply *C* to all nodes in the network and mark those below a threshold as malicious and those above as benign.

DOP(a): Remove all positive pairs of edges between benign nodes.

DOP(b): Remove all negative pairs of edges between benign nodes.

DOP(c): Remove all pairs of edges between benign nodes where one edge is positive and the other is negative.

DOP(d): In a positive/negative pair between two benign nodes, remove the negative edge.

DOP(e): In a positive/negative pair between two benign nodes, remove the positive edge.

After this, we iteratively perform the following steps, given a subset *D* of these 5 decluttering operations.

Repeat until convergence or a certain number of iterations is reached

- Apply the decluttering operations in *D* to edges linking pairs of nodes in the graph.
- Recompute the centrality measures.

End-Repeat

We then ran extensive tests on a real world Slashdot signed network data set. In this network, we showed that the best results are obtained when our Troll Identification Algorithm (TIA) uses the Signed Eigenvector Centrality with decluttering operations *a* and *e*. In this setting, we retrieve more than twice as many trolls as Negative Rank does in the bottom 96 ranked users. The running time of this algorithm is less than 2 minutes, which is quite reasonable – and more than 27 times faster on the original network and 35-50 times faster on the sub-networks as compared to using the best centrality measure (Negative Rank) used in past work.

Simply put, our TIA algorithm is both more accurate and faster than past work to identify such malicious users on signed networks like Slashdot.

V. EXPERIMENTS ON MALICIOUS USER BEHAVIORS IN ENTERPRISES

We also developed an experimental framework called BAIT (Behavioral Analysis of Insider Threat) to predict which behavioral cues can help predict insider threat. In particular, BAIT represents one of the first studies of real-world users attempting to compromise a system. BAIT contains bootstrapped algorithms that try to learn separators between malicious insiders and honest users for an environment with the following properties:

- Honest and malicious insiders' data is gathered from real human players. We took a number of steps (detailed later) to ensure that only high-quality subjects dedicated to their mission were considered in our study.

- The attacks are carried out by real humans similar to employees in an organization without regard to past attacks reported in the literature. In particular, our results are not biased by the possibility of both attackers and defenders studying past cases of insider attacks (which has biased many previous studies).

- Malicious insiders are directed to hide their actions.
- There is almost no labeled training data and the training data set is highly imbalanced.
- The system does not have any data on users' past behavior.
- The system may only observe the players' actions on a high-level and does not have any access to specific features of the documents which the players are handling or should handle.

We tested BAIT using a set of 795 users on Amazon Mechanical Turk and learned behaviors of real world users who both act normally and who are charged with exfiltrating data from a (synthetic) organization. To the best of our knowledge, this is the largest such study of insider threat ever using real humans. Players were told that they have to select tasks that they would perform in a high security facility on various topics. They do not actually perform these actions, but rather select them from a list of predefined possible actions. The players were told that they are required to gather information on several assigned topics (e.g. an analyst might be tasked to collect information on piracy incidents off the Horn of Africa), edit this information and send it (or hand it) to the topic's requester.

Events within BAIT play our over a period of 35 "rounds". The player may either be an honest worker or a malicious insider. Both types of workers receive a list of between 3 to 5 "classified" (synthetic of course) 5 topics. The players are told to gather information on each of these topics, edit this information and send it to the topic's requester. The players are told to assume that all the information that they are dealing with is classified (though in reality this is not the case). The players are given an additional topic ("personal interests"), to which they may attribute all actions not strictly related to their work. Such actions are a proxy for the fact that some analysts may check basketball data while on their classified machines or ballet performances, reflecting things in which they are interested. All players are told that the system is monitoring their actions and that they should make sure they are not discovered.

Malicious insiders are given an additional synthetic topic *t* (e.g. design plans for a new missile) and told that their job is to come up with actions that exfiltrate that data, while minimizing the probability of detection by the surveillance system and/or human cyber-security personnel. This type of worker is told that the topic *t* is his top priority and that he must try to hide/disguise any actions that he performs related to this topic. Note that work on the extra malicious topic is performed alongside the other tasks that the player performs. This simulates the behavior of malicious insiders, who perform both regular tasks and a small amount of malicious tasks. In every round of the game, each player may choose to perform one of the following 13 actions regarding each of the topics assigned to him.

1) Do nothing.

2) Fetch a document from the confidential internal database related to [a selected topic].

3) Ask a colleague to send a document related to [a selected topic].

4) Fetch a document from a CD / DVD related to [a selected topic].

5) Fetch a document from the USB drive related to [a selected topic].

6) Fetch unclassified (synthetic) data from the internet / email, related to [a selected topic].

7) Send an email with encrypted data related to [a selected topic] to the topic's requester.

8) Save data to a CD / DVD related to [a selected topic], and deliver it to the topic's requester.

9) Save data to the USB drive related to [a selected topic], and deliver it to the topic's requester.

10) Send unencrypted data related to [a selected topic] to the topic's requester via email/internet.

11) Print information related to [a selected topic], and hand the pages to the topic's requester.

12) Send encrypted data over the internet related to [a selected topic] to the topic's requester.

13) Edit information on [a selected topic].

Note that the number of actions we allow the user to perform is actually much greater than 13 because 12 of the actions are each parameterized by a topic. Thus, BAIT allows the user to select any of the $12T+1$ topics where T is the total number of possible topics. Each player needs to perform 35 actions (of which he may do the "nothing" action as many times as he wants). Every time a player performs an action related to a topic, that action is listed under the associated topic. The system does not see the topics, but only knows what actions were performed. We term the actions related to transferring data as "transfer actions". The transfer actions include: save to CD, save to USB, print and 3 “send” type actions: email, internet, unencrypted. For the three “send” actions, the system can also determine whether the actions were internal or external, resulting in 6 possible “send” type-destination combination actions. The internal destination “send” operations send information within the organization. The external destination send operations send data out of the organization, and can be used by either honest users (e.g. for following/communicating their personal interests) or by malicious users (e.g. to exfiltrate data). However the system does not know if the external destination operations were used by honest users for their personal reasons or by malicious users to send sensitive data to people outside the organization which is strictly forbidden.

The data available in our domain is generally composed of a very small amount of labeled data and large amounts of unlabeled data. This motivates the use of Semi Supervised Learning. The BAIT system includes a suite of seven algorithms that are built on top of two classical machine learning algorithms — Support Vector Machines [8] or SVM and Multinomial Naive Bayes [89]. These algorithms allow us to learn conditions that distinguish between behaviors of the malicious users and the benign ones.

In all SVM methods, we determine the best kernel by using cross validation on the labeled data. We considered linear kernels, polynomial kernels of degree 2 and 3, and the Radial Basis Function (RBF) kernel. We use cross validation on the labeled data also when considering the best method for dealing with imbalanced data. We consider oversampling of the minority group (the malicious insiders), under-sampling of the majority group (the benign users) and cost weight adjustments. We propose 5 methods for building the model. In methods 2-5 we use a confidence measure. The confidence is defined as the greatest distance from the separator hyper-plane computed by SVM.

*Algorithm BAIT*_L - *Use Only Labeled Data.* The BAIT_L algorithm is extremely simple and serves as a baseline. Given a set L of labeled data, it merely uses SVM to compute a separator hyperplane and then uses this as a classifier. Nothing particularly intelligent is done other than the use of SVM as a classifier.

Algorithm BAIT_LUMI - *Use Labeled Data +* I *Unlabeled Malicious Insider.*

The BAIT_LUMI algorithm is smarter than the baseline BAIT_L. It takes as input, a highly imbalanced and very small labeled data set L with very few labeled malicious players as well as an unlabeled data set U, and an integer $k \geq 0$.

It first uses the BAITL algorithm above to learn a separator using SVM on the labeled data. Next, it labels all users in U using the SVM learned. It then identifies the top k malicious users (Ltop) (i.e., the k users with the highest confidence). It then re-learns a new separator, using L ∪ Ltop as a training set. It then associates labels for all users in U - Ltop using the learned separator (of course, the labeled entries in Ltop keep their labels.) and denote it LU . Then it returns the separator learned from LU Ltop ∪ LU .

In short, BAIT_LU MI constructs a model by classifying the unlabeled data using an SVM learned on the labeled data, chooses the top k malicious insiders from this set in order to reduce the imbalance in the data between malicious and honest users, and then retrains and reclassifies. The idea of choosing the top k malicious users is that these users are the most likely to have been correctly classified in the first step.

Algorithm BAIT_LHUMI - *Use Labeled Data + Honest and Unlabeled Malicious Insider.* In the BAITLHUMI algorithm (Algo. 2), we extend the idea in BAITLUMI. As before, we first learn an SVM model M from the labeled data set. But instead of just looking at the top k users labeled as malicious by M in the unlabeled data set, we identify both the top k malicious users and the top l · k honest users and add them to the training set and then recalibrate the model M. The reason for adding only the top k malicious insiders but not adding a much larger number of honest users is to account for the imbalance in the data as in both the real-world and in our data set, the percentage of malicious insiders is very small.

Algorithm BAITI −LHUMI - *Use Labeled Data and Iteratively Update Unlabeled Data with Malicious User and Honest Users*.

The BAITI −LHUMI algorithm (Algo. 3) adapts the BAIT_LUMI algorithm by iteratively processing the unlabeled data with the most recently learned SVM model, then adds the most likely malicious user and a larger number of the most likely honest users back into the labeled data, and recalibrates. The most likely malicious/honest user is added back into the labeled dataset, one at a time. By adding each back one at a time, we hope to obtain better models, due to the fact that the model is recomputed after each new labeled user is captured.

Algorithm BAIT_MI_LHUMI - *Use Labeled Data and Iteratively Update Unlabeled Data with Malicious User and Honest Users*.

In the BAIT_MI_LHUMI algorithm (Algo. 4), we extend the BAIT_LHUMI algorithm in a different way. We first learn an SVM model M from all of the labeled training data L. We then classify the unlabeled data U using this learned model. We then randomly choose one malicious user from the top r malicious users in U for some value of r and label this person a malicious user — this is different than what occurs in the BAIT_LHUMI algorithm. Then, as in BAIT_LHUMI , we choose the top l honest users in U. These l + 1 users (l honest plus one malicious insider) are then added back to the labeled data set and this process is iterated within the inner loop of the algorithm until all U is labeled. The entire inner loop is executed J times for some J > 0 in order to smooth out some of the the random choices made in the computation. After iterating this J times, we get a total of J groups of malicious insiders. We label the s feature vectors which appear most frequently in these groups as malicious insiders and the others as honest and retrain the SVM using both this data and the *original* labeled data (L).

*Naive Bayes (*BAIT_NB *) Algorithm*.

The BAITNB algorithm is very simple — we merely applied the classical Multinomial Bayes algorithm on the labeled data and used that classifier to label the unlabeled data.

*Bayes with Iterative Probability Update (*BAIT_BIPU *) Algorithm*.

This algorithm (Algo. 5) is based on Expectation- Maximization (EM). Leveraging the idea presented by Nigam, McCallum and Mitchell [90], we first built a classifier model using the labeled data and then we improved the model iteratively using the unlabeled data together with the probabilities provided in the most recent iteration. This is executed iteratively till the model converges.

We developed various types of **features** based on the key indicators described earlier. All these features were chosen without observing the unlabeled/test data. Some features count the number of times an action was performed — which we term "basic" features. Some features are derived from the "basic" features using linear combinations, and some are derived by the division of one derived feature by another. The full list of features is given below:

• 16 basic features : The number of times each action was performed by the player, where the 3 sending actions are each split into internal and external sending.

• 12 features derived from sending actions:

- There are 3 types of "send" actions. For each of these types we define a feature that is the number of all internal sends (for this type) divided by the total number of "send" actions (internal and external for this type). This provides 3 features, one for each type.
    - Sum of all "fetch" actions (fetch from internal DB, ask colleague for documents, fetch a document from CD, fetch a document from USB).
    - Sum of all "transfer" actions.
    - Sum of all "send" actions.
    - Sum of all "send" actions divided by the sum of "transfer" actions.
    - Sum of all external "send" actions.
    - Sum of all external "send" actions divided by the sum of internal and external "send" actions.

- Sum of all “transfer” actions that are not “send” actions (save to CD, save to USB and print).
- Sum of all “unsuspicious fetch” actions. We define “unsuspicious fetch” actions as fetching classified data that does not involve other people.
- Sum of all “unsuspicious fetch” actions (de- fined in the previous item) divided by the sum of all “fetch” actions. When feeding these features to the SVM algorithms, we normalize the features to be between −1 and 1.

We tested the BAIT framework on Amazon’s Mechanical Turk service (AMT). The total number of subjects who participated in all the experiments included 795, all from the USA of whom 49.8% were females and 50.2% were males. The subjects’ ages ranged from 18 to 68, with a mean of 33.2, a standard deviation of 10.1 and median of 30. 7.7% of the subjects had either a PhD or a masters degree, 45.7% had a bachelor’s degree, 46.3% of the subjects had a high-school education and only 3 (0.36%) subjects had only completed junior high school.

In the labeled data-set, we ensured that exactly 7 subjects were malicious insiders, and in the unlabeled data-set and in the test data-set, each player had a 7% chance of being a malicious insider. The insiders were explicitly told to hide their actions.

We recruited only subjects with a history of at least 97% approval rate on AMT, and who had completed at least 60 tasks on AMT in the past. We further had all subjects pass a quiz ensuring that they were all diligent in following the instructions. In addition to all this, since the framework is an “open” and unusual one, we expected some subjects to have difficulty understanding and engaging with the system. Therefore, after the subjects completed their 35 rounds, they were asked whether they understood the instructions and whether they acted similarly to how they would have acted in real life. We removed 101 (12.7%) subjects who either indicated that they did not understand the instructions properly and/or indicated that they just picked random actions. This ensured that the subject pool was even more serious.

After removing these subjects there were a total of 100 labeled feature vectors (each feature vector contained data from a single subject over 35 rounds), of which 7 were labeled as malicious insiders and 93 as honest players, 394 unlabeled feature vectors and 200 feature vectors to use for testing.

We ran all 7 algorithms described above on our data. We first describe results for the SVM algorithms. We used the following settings: we set k = 11 since we assumed that the unlabeled data was sampled from the same distribution as the labeled data. Because there are 7 (of 100) malicious insiders in the labeled data, we know with a confidence of 95% that there are at least eleven malicious insiders. We sets = 28 since we assumed 7% of malicious insiders in the unlabeled data, since we know that we have that proportion in the labeled data. We set l = 93/7 = 13.3, r = 3,J = 50.

Cross validation on the labeled data encouraged us to use a linear kernel and to set the cost weight function at 0.01 for the honest feature vectors and 0.1 for the malicious ones. We used all the features described in 3.4 for the SVM models. The results are described in table 2. As presented in the table, all methods except BAIT_L reached a similar recall of 0.5 − 0.6, however, the precision rate varied from 0.17 in BAIT_LUMI to 0.3 in BAIT_MI_LHUMI, resulting in an f-measure of 0.4 (in BAITMI−LHUMI).

In all algorithms the recall rate was higher than precision. Indeed, when considering insider threat, recall is more important as it implies that more insiders will be detected, even if the precision rate is relatively low, which implies that more (possibly honest) people will be interviewed and required to explain their actions.

We also ran Multinomial Naive Bayes on our data. The actions played in the game can be seen as analogous to words, assuming internal and external sends are seen as different words. This motivated us to test Multinomial Naive Bayes on the set of “basic” features. Unfortunately the “derived” features cannot naturally be used in Multinomial Naive Bayes. We wanted to compare the Multinomial Naive Bayes results to SVM, However SVM did not succeed in building a model solely using the “basic” features. The results are described in the table below. Interestingly, BAIT_BIPU does not seem to work well with imbalanced data. Therefore we can conclude that if it is possible to add additional smart features (the derived sending actions), it is better to use the BAIT_MI_LHUMI method. However, if it is not possible to add additional smart features, it is better to use BAIT_NB (which is based on Multinomial Naive Bayes). When considering only the labeled data, the Naive Bayes method without the additional smart features seems to perform at least as well as the SVM with the additional smart features.

Last but not least, we briefly note that we tested the use of one-class SVMs that have been used in prior work on our data. The ocSVMs performed very poorly on our highly imbalanced data with very little past history. Consequently we abandoned this approach.

At the end of the day, we learned the following important results that enable us to find malicious insiders within enterprises.

- Moreover, we learned differences between behavior of real malicious users and that of benign users. Specifically we learned that:
    - The malicious players were more active and chose to do ”nothing” significantly (p <0:01) less times (3:77) than the benign players (6:52).

- The malicious players fetched significantly (p < 0:05) more ”sensitive information (9:8) than the benign players (8:26).
- The malicious players appeared to save more data to CD/DVD (1:2) and USB (2:225) compared to the benign players CD/DVD (1:07) and USB(1:82). However, these differences were not found to be statistically significant in our study. These findings vary slightly from findings in previous studies which were found to be statistically significant.
- The malicious players edited the data slightly less (3:77) compared to the benign players (3:88). However, these differences also were not found be statistically significant. These findings also vary from findings previously reported in the literature which showed the difference to be statistically significant.
- In addition, we found the following statistically significant differences between the malicious and the benign players. No similar findings have been reported in prior work.
    - The malicious players sent significantly (p <0.001) more information out of the organization with an average of (2:92) times, in contrast to the benign players (0:68).
    - The malicious players fetched significantly (p = 0:05) less unclassified data (2:87) in contrast to the benign players (3:40).

These behaviors were coded as conditional probability behavior rules and formed our Conditional Probability Knowledge Base. As we can see above, such rules may be automatically learned from historical data (training set) not about individual users, but about users focused on certain topics.

Simply put, some of these findings are extremely novel and cast important light on the behavior of real world insiders as compared to synthetic insiders injected somehow by an artificial process.

VI. COLLABORATION GRAPHS

Many types of network data can be used to identify insider threat. For instance, we have already shown that users on Wikipedia generate a graph in which these users collaborate together to serve a public good. However, some users try to destroy this goal by engaging in acts of vandalism. Wikipedia is an example of a collaboration graph – an area that subcontractor Boston Fusion developed further.

The purpose of the collaboration graph is to provide a network construct that focuses on the most significant communication activities for each individual in a network. By focusing on the most significant communications, we highlight the most "normal" network behavior for each entity, while also managing the size of the original observation network data set. We will use the collaboration graph to compute sets of features that describe the typical collaboration behavior of each individual in the network. The collaboration graph is derived graph structure, which we can build for either directed or undirected graph data sets, although the collaboration graph itself is a multi-edge, directed graph. The only parameter for the collaboration graph is *N*, indicating that the collaboration graph should include the top-*N* most significant communications for each user.

We construct the collaboration graph for a specified interval of time, during which all communications become additional link evidence. We do not, however, explicitly track time within the collaboration graph itself. To consider how collaboration behaviors evolve over time, we construct sequences of collaboration graphs, and use features or descriptors of the collaboration graph to compare sequences of graphs (for example, see below for a discussion on wavelet decomposition techniques).

To support the ADEN system directly with the collaboration graph, we generated a secondary set of numerical-based features to describe the structure and information flow/brokerage by individuals based on the feature graph characteristics. These roles describe communication patterns for inter- and intra-departmental communication and information flow, leveraging Gould's work in defining such roles, which could be helpful in understanding how people collaborate within and across departments in the Vegas data. The following features, based on Gould's theory of information brokerage in transactional networks and general network analysis, can be run on either the raw communication graph or the collaboration graph:

- Coordinator Score, (degree to which an individual coordinates communication across the department)
- Gatekeeper Score, (degree to which an individual brokers incoming communications to the department)
- Itinerant Broker Score, (degree to which an individual brokers internal communications for a department from outside that department)
- Liaison Score, (degree to which an individual brokers communications between two external department)
- Representative Score, (degree to which an individual brokers outgoing communications to the department), and

- Routing Score (degree to which an individual is relied on for facilitating communications between two other individuals).
- Clustering Coefficient
- InDegree - total number of emails received by the user
- Number of Senders
- OutDegree - total number of emails sent by the user
- Number of Recipients

Additional collaboration features can form the basis for a two-stage clustering algorithm, which would identify the individual(s) that are most likely to interact with a selected individual. ADEN could use these features to (1) determine which individuals are demonstrating shifts in their social/collaboration interactions, and (2) what level of shifting is significant. These features include:

- Mutual Contacts - for two users, find the contacts they have in common
- Similar Connections - for a user U, find users who interact with U's contacts, as well as the number of contacts they have in common, ordered by the number of contacts in common
- Skip Level Associates - for user U, find users who interact with U's contacts but who do not directly interact with U
- Top N Users - the top N senders and or recipients
- Top N Senders
- Top N Receivers

These features can help the system user identify:

- Who the most likely collaborators are in the upcoming time interval, based on past interactions;
- What the expected “skip-level” communications are, to detect whether individuals appear to be reaching out beyond normal reporting or communication chains;
- What the expected social network structure is for the most-likely collaborators, based on diversity of communicators and through the clustering coefficient on a reduced collaboration graph.

Boston Fusion generated a technical report describing the construction of the collaboration graph for directed and undirected graphs, and the features referenced above.

The implementation of the collaboration graph enables the user to select the “depth”, *N,* of the collaboration graph, which indicates that the graph should only include the top N weighted edges for each node. Note that we do not need to recompute the graph for each specified value of *N*, but regard edges as virtually “pruned” for subsequent computations, allowing the system to efficiently consider graphs for other values of *N* without reinitiating the construction procedure.

Boston Fusion deployed the collaboration graph constructor and numerical feature generators to the Vegas lab. We generated collaboration graphs on data in the Vegas lab, and generated features on these collaboration graphs. We also identified an approved set of feature values to bring back to Boston Fusion for additional testing and development, including clustering coefficient, number of skip level users, and number of similar users, for all users from July 2012-April 2013.

VII. WAVELET BASED TIME SERIES AND TREND ANALYTICS

Boston Fusion conducted investigations into supplemental methods for modeling behaviors observed in organization-based social networks with the objective of determining if we can establish the “physics” of how individuals behave in an organization, and identify the activities they conduct that may give them the most latitude in how they use time or complete tasks. For some activities, such as work arrival and login times, behaviors may be harder to alter without being noticed, whereas others, such as the balance of time spent on websites or email, may be harder for others to observe, and would allow for greater variability in behavior. We believe that anomalies within both sets of activities are ultimately valuable, but that each would potentially point to a different set of potential threat or risk types.

Our goal was to identify when individuals do things in consistent manners because there are rules (i.e., mandated to be completed in a certain way) or habits (reflecting the personal routines an individual establishes to complete their expected tasking). Our hypothesis is that individuals, when deviating from normal behavior, are more likely to break habits than to break rules, since rule-breaking is more detectable and more likely to carry consequences. This information will enable us to determine when individuals are changing from their normal routines, but still attempting to appear compliant within the user population.

Wavelet decomposition is similar to Fourier transforms in that both techniques decompose a target function into constituent elements. Fourier transforms capture periodic behavior in a set of data. They indicate the frequencies that are present, and the amplitudes of their constituent functions (sinusoids). The downside is that the Fourier transform completely forgets localization in time. In contrast, wavelet decomposition can capture this desired localization. Instead of examining just shifts of a window function, the wavelet transform allows shifting and scaling, which lets it vary sensitivity with frequency (at a cost of inversely varying the time resolution). To explore the approach further, we:

- Explored and implemented both continuous and discrete wavelet approaches, employing the MATLAB Wavelet Toolbox in addition to other MATLAB code, to determine the normal "ebb and flow" of collaboration patterns that may be detected with wavelet decomposition techniques.

- Built the preliminary infrastructure for processing Vegas data. This infrastructure helped us identify some previously undiscovered issues with our internal data, which we resolved.

- Ran tests on sample data, which we derived from observations in a social media forum data.

- Processed initial set of Vegas metrics data (clustering coefficient, # similar users, # skip levels) through time series analysis software.

- Analyzed characteristics and features of Vegas metrics to determine applicability to more advanced wavelet analysis.

- Investigated correlations between Vegas metric data, specifically the # similar users and # skip levels. Determined that data is highly correlated, and considered the implications to future data collections.

We developed a technical report describing Wavelet decomposition, and how this technique can be used to characterize business rhythms in the ADAMS data.

VIII. ANALYST EXPLANATION ENGINE

We automatically assign tags to classify different log events, and like a typical tag cloud, the size of a tag in the browser indicates the number of events with that tag. However, like faceted browsing, the user can select a tag as a constraint, and get a filtered view that only considers events that match the constraint. Any number of tags can be added to the constraint in order to create a more specific filter, and at any time a tag can be removed to broaden the query. In this way the user can explore the data freely, discovering interesting patterns and investigating them in numerous ways.

The primary link with the rest of ADEN was the indicator outputs. The Detection Engine provides a list of anomalies, where each anomaly has a user id, week, and indicator variable. The Explanation Engine was initially designed to work with the indicators: "After Hour Activity," "Employment Termination," and "File Theft." Using the semantics of these indicators, the Explanation Engine tagged individual log events that are likely to contribute to each indicator (note, the Detection Engine does not output information at this fidelity). For example, after hour activity could be any user event in the indicated time period that happened between 8pm and 6am. We also included a tag called "Any Indicator" which subsumed all of the indicators. Using this tag, the analyst could quickly focus on users and events that might be anomalous.

To organize the exploration, we divided the display into a set of tabs: User, Time, Access, Document, Networking and Indicator. Each tab has a set of "blocks" that contain tags describing values for a particular feature. For example, the Time tab has blocks for month/year, week, day-of-week, time-of-day, and hour. Note that in some cases, tags are intentionally redundant (e.g., hour implies time-of-day). This is an important feature, as such tags allow the user to see patterns at different levels of abstraction. As long as the tags co-occur, a tag can be combined with tags from any block or tab to form a new filter.

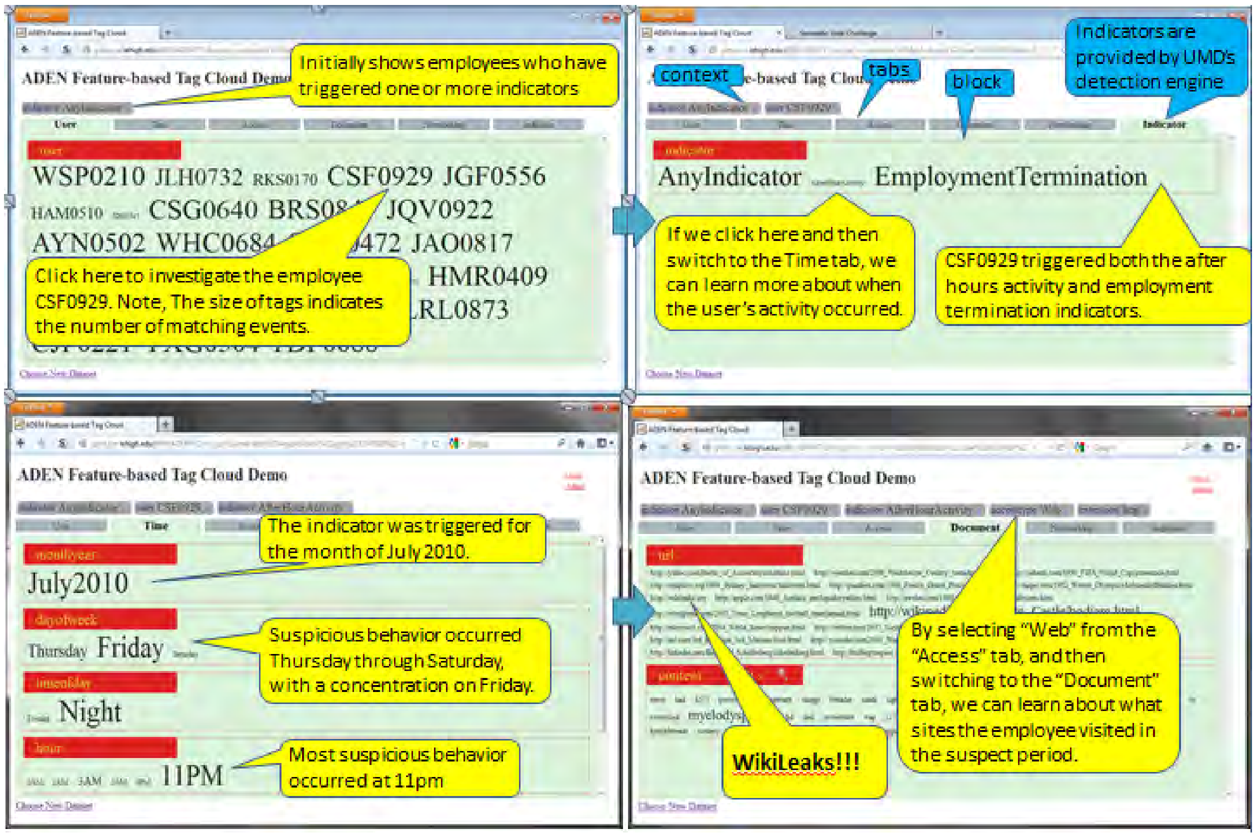
Our system also provides a comparison feature. For any filter, the analyst can choose to enter compare mode, which splits the screens in to two columns each initialized with the same filter. The user can then choose to add or remove tags from either filter, thereby indicating a baseline and a comparison. Tags in each column are displayed based on the relative frequency of events within the filter. Tags that are more likely in events in the baseline filter are in blue, and the larger they are, the bigger the difference between the baseline and the comparison. Tags that are more likely in the comparison filter are green. Using this interface, the analyst can compare along any dimension. How does a particular user's activity in a month compare to all other users in the same month? How do the contacts of users who visit a particular website compare with the general population? Do users with large attachments send e-mails at different times than other users?

The challenge in building such a system is to make it responsive. Our key strategy was to use inverted indices, which are a data structure that enables scalable web search. For our purposes, a log event was treated as a document, and the tags we assigned to it as the terms. The inverted index then has a list of documents for each term and determining the size of a tag is simply done by counting the intersection of the document list of that tag with all tags in the filter. Still, to compute the information to display a given screen, many such queries are needed, at least one for each tag shown on the screen. One issue is finding tags with non-zero intersections, and a naïve approach issues many queries that do not add a tag to the screen. In order to optimize performance, we developed two strategies. Conditional Instance Processing (CIP) is used when the number of matching instances is small, and determines which tags to consider by first fetching these instances. Precomputed Candidate Set (PCS) relies on a data-structure that determines which tags co-occur on at least one event. This can be thought of as a sparse matrix. We conducted experiments using simulated queries, and determined that a system that used both CIP and PCS performed better than either one alone, with response times under 1 second 98.6% of the time using the CERT-generated synthetic data. The worst response time out of 5000 random queries was 5.5 seconds. The co-occurrence index improves runtime performance, but there is a pre-computation cost. If we create the index, it take 6.32 hour to build all indices for 10 million log events, while it takes less than 20 minutes to build just the event index for the same number of events. However, we believe it is elatively straightforward to distribute index building, so these times are not necessarily prohibitive.

Months 6 through 15 were primarily based on investigating the capabilities of the Explanation Engine using the synthetic log file data provided by CERT, particular r3v1, r4v2 and r6. Subsequently, efforts turned to how the system would interact with a SureView installation. We began by loading the synthetic data into an Oracle database with the SureView schema, and adapted the index building process to use queries over this database. Due to the number of results, our queries are first constrained to three month periods and fetch only 10,000 records at a time, which was determined experimentally to provide the best runtime while not causing memory issues.

After verifying that the process worked on our local database to our satisfaction, we were ready to perform tests on the Vegas dataset, contributed by Raytheon. With the help of David Stein, we ran various programs on this dataset and collected summary results from months 24 to 30. This dataset was 12x bigger than the synthetic datasets we worked with. Due to restricted access and limited export of products, debugging this was quite challenging. Using the combined CIP/PCS strategy, we found that average response time ranged from 3 to 6 seconds as the size of the contextual constraint increased. Display pages where the block was e-mail related were the fastest, while blocks about the URL (with 13 million possible URLs) were the slowest. We also compared two strategies for computing co-occurrence, and found that the Field Cache strategy was 4x faster than the Event-Based strategy. This strategy works by taking a tag at a time, finding all instances with that tag, and then using Lucene's field cache feature to efficiently find all tags for each of these instances.

Selected Screenshots of the explanation tool are shown below.


ADEN Feature-based Tag Cloud Demo
Initially shows employees who have triggered one or more indicators
WSP0210 JLH0732 RKS0170 CSF0929 JGF0556
HAM0510 CSG0640 JQV0922
AYN0502 WHC0684 JAO0817
HMR0409
RL0873
Click here to investigate the employee CSF0929. Note, The size of tags indicates the number of matching events.
context
tabs
block
Indicators are provided by UMD's detection engine
AnyIndicator EmploymentTermination
If we click here and then switch to the Time tab, we can learn more about when the user's activity occurred.
CSF0929 triggered both the after hours activity and employment termination indicators.
The indicator was triggered for the month of July 2010.
July2010
Thursday Friday
Suspicious behavior occurred Thursday through Saturday, with a concentration on Friday.
Night
Most suspicious behavior occurred at 11pm
3AM 11PM
By selecting "Web" from the "Access" tab, and then switching to the "Document" tab, we can learn about what sites the employee visited in the suspect period.
WikiLeaks!!!

IX. STUDENTS/POSTDOCS INVOLVED

Postdocs: Dr. Michael Ovelgonne, Dr. Rami Puzis, Dr. Francesca Spezzano

PhD Students: Mr. Chanhyun Kang, Mr. Noseong Park, Mr. Manish Purohit, Xingjian Zhang, Dezhao Song, Sambhawa Priya, Yang Yu, Zachary Daniels.

X. PAPERS

1. Zhang, X., Song, D., Priya, S., Daniels, Z., Reynolds, K., & Heflin, J. (2014). Exploring Linked Data with contextual tag clouds. Web Semantics: Science, Services and Agents on the World Wide Web, 24, 33-39.
2. Zhang, X., Song, D., Priya, S., & Heflin, J. (2013). Infrastructure for efficient exploration of large scale linked data via contextual tag clouds. In *The Semantic Web–ISWC 2013* (pp. 687-702). Springer Berlin Heidelberg.

3. O'Donovan, F. T., Fournelle, C., Gaffigan, S., Brdiczka, O., Shen, J., Liu, J., & Moore, K. E. (2013, July). Characterizing user behavior and information propagation on a social multimedia network. In *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on* (pp. 1-6). IEEE.
4. Azaria, A., Richardson, A., Kraus, S., & Subrahmanian, V. S. (2014). Behavioral Analysis of Insider Threat: A Survey and Bootstrapped Prediction in Imbalanced Data. *Computational Social Systems, IEEE Transactions on*, *1*(2), 135-155.
5. Kumar, Srijan, Francesca Spezzano, and V. S. Subrahmanian. "Accurately detecting trolls in Slashdot Zoo via decluttering." *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*. IEEE, 2014.
6. Kumar, Srijan, Francesca Spezzano, and V. S. Subrahmanian. "VEWS: Vandal Early Warning System for Wikipedia".